Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Системне програмування

Лабораторна робота №9

«Використання функцій API Win32 у програмах на асемблері»

Виконав:

студент групи ІО-82

Шендріков Є. О.

Залікова № 8227

Перевірив Порєв В. М.

Київ – 2020

Мета

Навчитися використовувати у програмах на асемблері функції Windows динамічного виділення пам'яті та запису файлів.

Завдання

1. Створити у середовищі MS Visual Studio проект з ім'ям **Lab9**.
2. Написати вихідний текст програми згідно варіанту завдання. У проекті мають бути головний файл **main9.asm** та модулі **module** (за необхідності) та модуль **longop** попередніх робіт.
3. У цьому проекті кожний модуль може окремо компілюватися.
4. Скомпілювати вихідний текст і отримати виконуємий файл програми.
5. Перевірити роботу програми. Налагодити програму.
6. Отримати результати – файл числових значень згідно варіанту завдання.
7. Проаналізувати та прокоментувати результати, вихідний текст та дизасембльований машинний код програми.

Варіант завдання

Запрограмувати на асемблері запис у файл масиву значень факторіалу n! (n від 1 до n_{max}). Вказування імені файлу у стандартному діалоговому вікні (функція GetSaveFileName). Для кожного студента своє значення n_{max}

$$\mathbf{n_{max} = 30 + 2 \times H}$$, де H – це номер студента у журналі.

H = 25 => n_{max} = 30 + 2*25 = 80

Мій варіант

№ варіанту	Варіант циклу факторіалу	Код результату	Масиви для даних підвищеної розрядності	Масив-буфер для імені файлу
25	1	десятковий	динамічні	статичний

Текст програми

main9.asm

```
.586
.model flat, stdcall

option casemap :none

include \masm32\include\kernel32.inc
include \masm32\include\user32.inc
include \masm32\include\windows.inc
include \masm32\include\comdlg32.inc

include longop.inc
include module.inc

includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\user32.lib
includelib \masm32\lib\comdlg32.lib   ; comdlg32.lib - діалогове вікно вказування імені файлу


.data
	x dd 1
	hFile dd 0

	myFileName db 256 dup(0), 0 ;буфер для імені файлу

	pResult dd 0
	pBuf dd 0

	decCode db 1024 dup(0) , 13,10, 0
	line db " ",13,10, 0

	n dd 0
	nm dd 80
	pRes dd 0

.code

	MySaveFileName proc
		LOCAL ofn : OPENFILENAME
		invoke RtlZeroMemory, ADDR ofn, SIZEOF ofn ; спочатку усі поля обнулюємо
		mov ofn.lStructSize, SIZEOF ofn
		mov ofn.lpstrFile, OFFSET myFileName
		mov ofn.nMaxFile, SIZEOF myFileName
		invoke GetSaveFileName,ADDR ofn ; виклик вікна File Save As
		ret
	MySaveFileName endp


	main:

		call MySaveFileName
		cmp eax, 0  ; перевірка якщо у вікні FileSaveAs було натиснуто Cancel, то EAX = 0
		je @exit
			; відкриття або створення файлу якщо немає - CreateFile
			invoke CreateFile,  ADDR myFileName,
                                            GENERIC_WRITE,
                                               FILE_SHARE_WRITE,
                                               0, CREATE_ALWAYS,
                                               FILE_ATTRIBUTE_NORMAL,
```

```
                    ; GENERIC_WRITE і тд - константи з windows.inc
                    cmp eax, INVALID_HANDLE_VALUE ; INVALID_HANDLE_VALUE - якщо значення не
дорівнює цьому то доступ дозволено
                    je @exit ;доступ до файлу неможливий

                    mov hFile, eax
                    invoke GlobalAlloc, GPTR, 1024

                    mov pResult, eax
                    add eax, 512
                    mov pBuf, eax
                    mov dword ptr[eax], 1  ; val = 1

                    ; обчислення факторіала
                    @cycle:
                           inc dword ptr[n]  ; n = n + 1
                           mov eax, dword ptr[n]
                           cmp eax, nm  ; 80!
                           jg @endf

                           push pResult
                           push pBuf
                           push x
                           call Mul_N_x_32_LONGOP  ; Result = val * n

                           push pResult
                           push offset decCode
                           push 16
                           push 120
                           call StrToDec_LONGOP

                           invoke lstrlen, ADDR decCode
                           invoke WriteFile, hFile, ADDR decCode, eax, ADDR pRes, 0

                           invoke lstrlen, ADDR line
                           invoke WriteFile, hFile, ADDR line,  eax, ADDR pRes, 0

                           inc x

                           push pResult
                           push pBuf
                           push 16
                           call COPY_LONGOP ; val <- Result

                           jmp @cycle

             @endf:
             invoke GlobalFree, pResult
             ; файл обовязково треба закрити
             invoke CloseHandle, hFile

             @exit:
                    invoke ExitProcess, 0
end main
```

```
.586
.model flat, c

.data

	x dd 0h
	x1 dd 0h
	x2 dd 0h

	b dd 0
	fractionalPart db ?

	two dd 2
	buf dd 80 dup(0)
	decCode db ?

	buffer dd 128 dup(?)

.code

Mul_N_x_32_LONGOP proc
	push ebp
	mov ebp, esp

	mov esi, [ebp + 16]
	mov edi, [ebp + 12]
	mov ebx, [ebp + 8]
	mov x, ebx

	mov ecx, 8
	xor ebx, ebx
	@cycle1:
		mov eax, dword ptr[edi + 8 * ebx]
		mul x
		mov dword ptr[esi + 8 * ebx], eax
		mov dword ptr[esi + 8 * ebx + 4], edx

		inc ebx
		dec ecx

		jnz @cycle1

	mov ecx, 8
	xor ebx, ebx

	@cycle2:
		mov eax, dword ptr[edi + 8 * ebx + 4]

		mul x

		clc
		adc eax, dword ptr[esi + 8 * ebx + 4]
		mov dword ptr[esi + 8 * ebx + 4], eax
		clc
		adc edx, dword ptr[esi + 8 * ebx + 8]
		mov dword ptr[esi + 8 * ebx + 8], edx

		inc ebx
		dec ecx

		jnz @cycle2
```

```
	pop ebp
	ret 12

Mul_N_x_32_LONGOP endp

DIV_LONGOP proc

	push ebp
	mov ebp, esp

	mov esi, [ebp + 20]  ; number
	mov edi, [ebp + 16]   ;integer
	mov ebx, [ebp + 12]  ;fractional
	mov eax, [ebp + 8]   ; bytes
	mov x, eax

	push ebx
	xor edx, edx
	mov ecx, x
	dec x
	mov ebx,x
	@cycle :
		push ecx
		mov ecx, 10
		mov eax, dword ptr[esi + 4 *  ebx]

		div ecx
		mov fractionalPart, dl
		mov dword ptr[edi + 4 * ebx], eax
		dec ebx
		pop ecx
		dec ecx

		jnz @cycle

	pop ebx
	mov al, fractionalPart
	mov byte ptr[ebx], al
	pop ebp
	ret 16

DIV_LONGOP endp

StrToDec_LONGOP proc

	push ebp
	mov ebp, esp

	mov esi, [ebp + 20] ;str code
	mov edi, [ebp + 16] ;dec code
	mov eax, [ebp + 12]
	mov x1, eax      ; number of dd
	mov eax, [ebp + 8]
	mov x2, eax      ; bytes on screan


	push esi
	push edi


	push esi
	push offset buffer
```

```
push x1
call COPY_LONGOP

pop edi
pop esi

mov b, 0

xor ecx, ecx
xor ebx, ebx
@cycle:
	push ecx
	push edi

	push esi
	push offset buf
	push offset decCode
	push x1
	call DIV_LONGOP

	pop edi
	mov ebx, b
	mov al, byte ptr[decCode]
	add al, 48
	mov byte ptr[edi + ebx], al

	xor ecx, ecx
	@cycleInner:
		mov eax, dword ptr[buf + 4 * ecx]
		mov dword ptr[esi + 4 * ecx], eax
		mov dword ptr[buf + 4 * ecx], 0
		inc ecx
		cmp ecx, x1
		jl @cycleInner

	pop ecx
	inc ecx
	inc b
	cmp ecx, x2
	jl @cycle

mov ebx, x2
mov eax, x2
xor edx, edx
div two
mov x2, eax
dec ebx
xor ecx, ecx
@cycle1:

	mov al, byte ptr[edi + ecx]
	mov ah, byte ptr[edi + ebx]
	mov byte ptr[edi + ecx], ah
	mov byte ptr[edi + ebx], al

	dec ebx
	inc ecx
	cmp ecx, x2
	jl @cycle1

push offset buffer
push esi
push x2
```

```
	call COPY_LONGOP


	pop ebp
	ret 16

StrToDec_LONGOP endp

COPY_LONGOP proc

	push ebp
	mov ebp, esp

	mov esi, [ebp + 16]
	mov edi, [ebp + 12]
	mov edx, [ebp + 8]


	mov ecx, 0
	@cycle:
		mov eax, dword ptr[esi + 4 * ecx]
		mov dword ptr[edi + 4 * ecx], eax
		inc ecx

		cmp ecx, edx
		jne @cycle

	pop ebp
	ret 12

COPY_LONGOP endp
end
```

module.asm

```
.586
.model flat, c
.code
;процедура StrHex_MY записує текст шістнадцятькового коду
;перший параметр - адреса буфера результату (рядка символів)
;другий параметр - адреса числа
;третій параметр - розрядність числа у бітах (має бути кратна 8)
StrHex_MY proc
 push ebp
mov ebp,esp
mov ecx, [ebp+8] ;кількість бітів числа
cmp ecx, 0
jle @exitp
shr ecx, 3 ;кількість байтів числа
mov esi, [ebp+12] ;адреса числа
mov ebx, [ebp+16] ;адреса буфера результату
@cycle:
mov dl, byte ptr[esi+ecx-1] ;байт числа - це дві hex-цифри
mov al, dl
shr al, 4 ;старша цифра
call HexSymbol_MY
mov byte ptr[ebx], al
mov al, dl ;молодша цифра
call HexSymbol_MY
mov byte ptr[ebx+1], al
mov eax, ecx
cmp eax, 4
```

```
jle @next
dec eax
and eax, 3 ;проміжок розділює групи по вісім цифр
cmp al, 0
jne @next
mov byte ptr[ebx+2], 32 ;код символа проміжку
inc ebx
@next:
add ebx, 2
dec ecx
jnz @cycle
mov byte ptr[ebx], 0 ;рядок закінчується нулем
@exitp:
pop ebp
ret 12
StrHex_MY endp
;ця процедура обчислює код hex-цифри
;параметр - значення AL
;результат -> AL

HexSymbol_MY proc
and al, 0Fh
add al, 48 ;так можна тільки для цифр 0-9
cmp al, 58
jl @exitp
add al, 7 ;для цифр A,B,C,D,E,F
@exitp:

ret
HexSymbol_MY endp
end
```

Результати роботи програми

```
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000002
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000006
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000024
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000120
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000720
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000005040
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000040320
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000362880
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000003628800
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000039916800
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000479001600
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000006227020800
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000087178291200
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001307674368000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000020922789888000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000355687428096000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000006402373705728000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000121645100408832000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000002432902008176640000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000051090942171709440000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001124000727777607680000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000025852016738884976640000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000006204484017332394393600000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000155112100433309859840000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000004032914611266056355840000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000001088886945041835216076800000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000030488834461171386050150400000000
```

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000008841761993739701954543616000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000265252859812191058636308480000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000008222838654177922817725562880000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000263130836933693530167218012160000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000008683317618811886495518194401280000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000295232799039604140847618609643520000000
000000000000000000000000000000000000000000000000000000000000000000000000000000010333147966386144929666651337523200000000
000000000000000000000000000000000000000000000000000000000000000000000000000000371993326789901217467999448150835200000000
000000000000000000000000000000000000000000000000000000000000000000000000000013763753091226345046315979581580902400000000
000000000000000000000000000000000000000000000000000000000000000000000000000523022617466601111760007224100074291200000000
000000000000000000000000000000000000000000000000000000000000000000000000020397882081197443358640281739902897356800000000
000000000000000000000000000000000000000000000000000000000000000000000000815915283247897734345611269596115894272000000000
000000000000000000000000000000000000000000000000000000000000000000000033452526613163807108170062053440751665152000000000
000000000000000000000000000000000000000000000000000000000000000000001405006117752879898543142606244511569936384000000000
000000000000000000000000000000000000000000000000000000000000000000060415263063373835637355132068513997507264512000000000
000000000000000000000000000000000000000000000000000000000000000002658271574788448768043625811014615890319638528000000000
000000000000000000000000000000000000000000000000000000000000000119622220865480194561963161495657715064383733760000000000
000000000000000000000000000000000000000000000000000000000000005502622159812088949850305428800254892961651752960000000000
000000000000000000000000000000000000000000000000000000000000258623241511168180642964355153611979969197632389120000000000
000000000000000000000000000000000000000000000000000000000012413915592536072670862289047373375038521486354677760000000000
000000000000000000000000000000000000000000000000000000000608281864034267560872252163321295376887552831379210240000000000
000000000000000000000000000000000000000000000000000000030414093201713378043612608166064768844377641568960512000000000000
000000000000000000000000000000000000000000000000000001551118753287382280224243016469303211063259720016986112000000000000
000000000000000000000000000000000000000000000000000080658175170943878571660636856403766975289505440883277824000000000000
000000000000000000000000000000000000000000000000004274883284060025564298013753389399649690343788366813724672000000000000
000000000000000000000000000000000000000000000000230843697339241380472092742683027581083278564571807941132288000000000000
000000000000000000000000000000000000000000000012696403353658275925965100847566516959580321051449436762275840000000000000
000000000000000000000000000000000000000000000710998587804863451854045647463724949736497978881168458687447040000000000000
000000000000000000000000000000000000000000040526919504877216755680601905432322134980384796226602145184481280000000000000
000000000000000000000000000000000000000002350561331282878571829474910515074683828862318181142924420699914240000000000000
000000000000000000000000000000000000000138683118545689835737939019720389406345902876772687432540821294940160000000000000
000000000000000000000000000000000000008320987112741390144276341183223364380754172606361245952449277696409600000000000000
000000000000000000000000000000000000507580213877224798800856812176625227226004528988036003099405939480985600000000000000
000000000000000000000000000000000031469973260387937525653122354950764088012280797258232192163168247821107200000000000000
000000000000000000000000000000001982608315404440064116146708361898137544773690227268628106279599612729753600000000000000
000000000000000000000000000000126886932185884164103433389335161480802865516174545192198801894375214704230400000000000000
000000000000000000000000000008247650592082470666723170306785496252186258551345437492922123134388955774976000000000000000
000000000000000000000000000544344939077443064003729240247842752644293064388798874532860126869671081148416000000000000000
000000000000000000000000036471110918188685288249859096605464427167635314049524593701628500267962436943872000000000000000
000000000000000000000002480035542436830599600990418569171581047399201355367672371710738018221445712183296000000000000000
000000000000000000000171122452428141311372468338881272839092270544893520369393648040923257279754140647424000000000000000
000000000000000000011978571669969891796072783721689098736458938142546425857555362864628009582789845319680000000000000000
000000000000000000850478588567862317521167644239926010288584608120796235886430763388588680378079017697280000000000000000
000000000000000061234458376886086861524070385274672740778091784697328983823014963978384987221689274204160000000000000000
000000000000004470115461512684340891257138125051110076800700282905015819080092370422104067183317016903680000000000000000
000000000000330788544151938641225953028221253782145683251820934971170611926835411235700971565459250872320000000000000000
000000000024809140811395398091946477116594033660926243886570122837795894512655842677572867409443815424000000000000000000
000000001885494701666050254987932260861146558230394535379329335672487982961844043495537923117729972224000000000000000000
000000145183092028285869634070784086308284983740379224208358846781574688061991349156420080065207861248000000000000000000
000011324281178206297831457521158732046228731749579488251990048962825668835325234200766245086213177344000000000000000000
000894618213078297528685144171539831652069808216779571907213868063227837990693501860533361810841010176000000000000000000
071569457046263802294811533723186532165584657342365752577109445058227039255480148842668944867280814080000000000000000000

Аналіз результатів

80! = 71569457046263802294811533723186532165584657342365752577109445058227039255480148842668944867280814080000000000000000000

Як бачимо кожен рядок файлу співпадає зі значеннями факторіалів від 1 до 80 у десятковій системі. Отже, результати програми повністю збігаються з теоретичними даними. Програма працює вірно.

Факторіал обчислюється аналогічно минулим роботам з цим завданням, за винятком того, що під час обчислення кожного наступного факторіалу після самого обчислення відбувається запис і перенос на новий рядок у файл.

Висновок

Під час виконання роботи я навчився працювати з функціями Win32 API, динамічною пам’яттю та створив програму, що обчислює факторіал чисел від 1 до 80 та записує їх у файл. Кінцева мета роботи досягнута.

Відповіді на контрольні запитання

1. **Що таке API Win32?**

WinAPI (також відомий як Win32, офіційно званий Microsoft Windows API) - це інтерфейс прикладного програмування, написаний на C Microsoft, щоб дозволити доступ до функцій Windows.

2. **Як викликати системну функцію ОС Windows?**

Щоб викликати системну функцію ОС Windows з API Win32 застосовують директиву INVOKE.

3. **Що таке хендл?**

Handle - дескриптор, тобто число, за допомогою якого можна ідентифікувати ресурс. За допомогою дескпріторов можна посилатися на вікна, об'єкти ядра, графічні об'єкти і т.д. Можна провести аналогію з масивом: у нас є набір ресурсів, а HANDLE - це індекс, який вказує на конкретний ресурс. В даній лабораторній, коли ми відкриваємо файл з ним зв’язується деякий ідентифікатор (handle). За допомогою нього можна викликати функції для запису у файл порцій інформації. Якщо значення handle не дорівнює INVALID_HANDLE_VALUE, то програмі дозволений доступ до файлу.

4. Як відкрити файл для запису даних?

Для цього можна скористатися функцією CreateFile. Ця функція відкриває старий або створює новий файл і записує handle файлу у регістр EAX. Після роботи з файлом необхідно його закрити, викликавши функцію CloseHandle.

5. Яка функція записує у файл?

Записати інформацію у файл можна за допомогою функції **WriteFile**.

6. Як створити динамічні масиви даних?

Для створення динамічного масиву можна використати функцію GlobalAlloc, яка належить до складу API Win32. Ця функція у залежності від параметрів виклику може повертати вказівник – адресу блоку пам'яті, що виділяється. Потім цей вказівник використовується для запису або читання потрібних даних. Коли динамічний масив стає непотрібним, його треба знищити за допомогою функції GlobalFree.